

Public Support - Support Request #12340

Analyze Memory Leaks

2020-10-20 13:30 - hidden

Status: Closed	
Priority: Normal	
Category:	
Customer: AUDI	Product Issue Numbers:
Department: EF	Affected Products: ADTF 3.8.0
Requester's Priority: Normal	Platform: Windows 10 64bit
Support Level: 2nd Level	Topic: ADTF::Common
Resolution: Solved Issue	FAQ Links:
Description	
Supportanfrage	
könnt ihr mir sagen, wie sich ADTF verhält, wenn kein Speicher mehr verfügbar ist? Wenn ein Filter beispielsweise ständig Memory-Leaks verursacht, stürzt ADTF dann irgendwann ab? Wenn ja, wird der Log-Output dann noch gespeichert, sodass ich den Absturz dann analysieren kann?	
Lösung	
Das kann man leider nicht allgemein beantworten, da prinzipiell jeder Filter jederzeit Speicher am heap anlegen kann und es dann darauf ankommt ob er das behandelt. Wenn er das über new in einer Funktion eines Filters macht müssten wir aber zumindest die std::bad_alloc exception abfangen und in einen Fehler umwandeln. Da aber auch das Speicher benötigt, kann man das nicht garantieren. alloc_sample (output_sample_data<>) etc. sollten den Fehler aber korrekt fangen.	
Ich würde aber vermuten, dass du letztendlich bei diesen Problemen mit einem Crash-Dump fast besser bedient bist.	
Das Log wird im Crash Handler von ADTF nicht gesondert behandelt, heißt das was bisher geschrieben wurde ist drin, mehr nicht. Hier kann eben leider der Schreibpuffer dazu führen, dass da nicht wirklich alles auf der Platte gelandet ist.	
Wenn man den lancher aber mit --log ... startet, dann wird das log file erzeugt und der Nutzer kann die Meldungen des Filters analysieren. Der Weg sollte für die künftigen Nutzer machbar sein.	

History

#1 - 2020-10-21 08:23 - hidden

- Project changed from Public Support to 11
- Topic set to ADTF::Common
- Customer set to AUDI
- Department set to EF
- Affected Products ADTF 3.8.0 added
- Platform Windows 10 64bit added

#2 - 2020-10-21 09:30 - hidden

- Status changed from New to In Progress

#3 - 2020-10-21 14:14 - hidden

Hi Claudia,

das kann man leider nicht allgemein beantworten, da prinzipiell jeder Filter jederzeit Speicher am heap anlegen kann und es dann darauf ankommt ob er das behandelt. Wenn er das über new in einer Funktion eines Filters macht müssten wir aber zumindest die std::bad_alloc exception abfangen und in einen Fehler umwandeln. Da aber auch das Speicher benötigt, kann man das nicht garantieren. alloc_sample (output_sample_data<>) etc. sollten den Fehler aber korrekt fangen.

Ich würde aber vermuten, dass du letztendlich bei diesen Problemen mit einem Crash-Dump fast besser bedient bist.

Das Log wird im Crash Handler von ADTF nicht gesondert behandelt, heißt das was bisher geschrieben wurde ist drin, mehr nicht. Hier kann eben leider der Schreibpuffer dazu führen, dass da nicht wirklich alles auf der Platte gelandet ist.

Grüße,

Martin

#4 - 2020-10-21 14:45 - hidden

Hi Martin,

ich habe das mittlerweile getestet. Der Filter sammelt alle seine Inputs in einem Vektor und der Arbeitsspeicher füllt sich dementsprechend. Nachdem der PC an seiner Grenze war, bekam ich bad_alloc exceptions und es wurde auch ein Crash Dump File gespeichert.

Ich weiß ehrlich gesagt nicht, wie ich dieses analysiere. Gibt es dafür spezielle Tools?

Aber ich vermute, dass die LOGs meines Filters nicht enthalten sind. Ein normaler User wird mit den Crash Dump Files sicher auch nichts anfangen können, daher war unsere Hoffnung, dass das LOG nach dem Absturz verfügbar ist.

Viele Grüße
Claudia

#5 - 2020-10-21 15:05 - hidden

Hi Claudia,

Claudia Haßmann wrote:

ich habe das mittlerweile getestet. Der Filter sammelt alle seine Inputs in einem Vektor und der Arbeitsspeicher füllt sich dementsprechend. Nachdem der PC an seiner Grenze war, bekam ich bad_alloc exceptions und es wurde auch ein Crash Dump File gespeichert. Ich weiß ehrlich gesagt nicht, wie ich dieses analysiere. Gibt es dafür spezielle Tools?

Einfach doppelklicken und Visual Studio öffnet sich. Kurzes googlen ergibt:
<https://docs.microsoft.com/en-us/visualstudio/debugger/using-dump-files?view=vs-2019>

Alternativ kannst du den adtf_launcher auch direkt in Visual Studio starten und debuggen, dann springt er auch gleich an die Absturzstelle.

Aber ich vermute, dass die LOGs meines Filters nicht enthalten sind. Ein normaler User wird mit den Crash Dump Files sicher auch nichts anfangen können, daher war unsere Hoffnung, dass das LOG nach dem Absturz verfügbar ist.

Im Crash Dump sind die Logs nicht enthalten, das ist ja wirklich der Dump des Prozess Images.

Hast du den adtf_launcher mit "--log-file mein_log_file.txt" gestartet? Dort sollte schon so einiges landen. Ansonsten kannst Du ihn auch mit --console starten, dann siehst du beim debuggen im Konsolenfenster noch die letzten Ausgaben.

Grüße,

Martin

#6 - 2020-10-22 08:00 - hidden

Hi Martin,

danke für die Hinweise.

Ich starte die Session immer aus dem Configuration-Editor heraus und gebe nichts Besonderes an. Die Console wird mir angezeigt und ich sehe auch meinen Output. Da ich aktuell weiß, dass ich den PC mit Memory Leaks überfordere, brauche ich auch nicht zu debuggen. Wir wollen aber sicherstellen, dass die künftigen Nutzer der Konfig auch auf solche Probleme hingewiesen werden. Mit dem Crash Dump File geht das nicht und die Console ist weg, wenn sich der PC aufgehängt hat. Heißt, wenn der Nutzer die Konfig über Nacht durchlaufen lässt, wird er am nächsten Tag einen abgestützten PC vorfinden, der ihm keinen Hinweis auf das Warum liefert.

Wenn man den launcher aber mit --log ... startet, dann wird das log file erzeugt und der Nutzer kann die Meldungen des Filters analysieren. Der Weg sollte für die künftigen Nutzer machbar sein.

Mein Problem ist damit gelöst und du kannst das Ticket schließen.

Vielen Dank
Claudia

#7 - 2020-10-22 08:20 - hidden

- Status changed from In Progress to To Be Closed

- Resolution set to Solved Issue

#8 - 2020-10-22 09:50 - hidden

- Project changed from 11 to Public Support

- Subject changed from *ADTF Verhalten bei Memory-Leaks* to *Analyze Memory Leaks*

- Description updated

- Private changed from *Yes* to *No*

#9 - 2022-01-17 08:17 - hidden

- Status changed from *To Be Closed* to *Closed*